Today’s class

- Antialiasing
- Hidden surface removal
- Texture mapping
Aliasing

- Result of changes in the object being drawn that occur faster than the pixel spacing
- Antialiasing techniques attempt to minimize the effect
- Attempt to have an intensity proportional to the amount of the pixel covered by the line
- Computations of pixel area covered are complex and time consuming
- If speed is important, live with the aliasing or double the resolution (doubles the scan convert time for a line, quadruples the number of pixels, is typically faster than antialiasing the original resolution)
Types of antialiasing

- Supersampling
- Area sampling
- Pixel phasing
Supersampling

- Displayed pixels cover finite areas
- In the algorithms we have discussed the intensity of each pixel has been determined by a single point on the object boundary
- By supersampling, we obtain intensity information from multiple points that contribute to the overall intensity of a pixel
Line segments

- Divide each pixel into a number of subpixels and count the number of subpixels that overlap the line path.
- The intensity level for the displayed pixel is set to a value that is proportionate to the count obtained.
- Can also consider finite width lines (with a width equal to a pixel) to get more levels of intensity.
Area sampling

- Pixel intensity is determined by calculating the areas of overlap of each pixel with the objects to be displayed
Pixel phasing

- Some raster systems can address subpixel positions within the screen grid.
- Smoothing is accomplished by moving pixel positions closer to the line path.
- Electron beam is shifted by a fraction of a pixel diameter, typically $\frac{1}{4}$, $\frac{1}{2}$, or $\frac{3}{4}$.
- This plots points closer to the true path of a line.
Intensity differences

- Diagonal lines appear to have lower intensities than horizontal or vertical lines with an equal number of pixels because the diagonal line is longer.
- Antialiasing techniques automatically adjust for this.
Antialiasing in OpenGL

- `glEnable (GL_POINT_SMOOTH);`
- `glEnable (GL_LINE_SMOOTH);`
- `glEnable (GL_POLYGON_SMOOTH);`
Z-buffer algorithm

- One of the simplest hidden surface removal algorithms
- A two-dimensional array is allocated, with one element for each pixel in the window
- The values stored in this array will be the \( z \) values of the faces currently displayed on the screen at the pixels
- As each face is processed, the pixels it covers are determined
Z-buffer algorithm (cont.)

- The z values for the face at these pixels are computed and compared to the values already stored in the z-buffer.
- If the new face’s z values are closer to the viewer (have smaller z values) then this face covers the previous face and its color is displayed on the screen (and its z values stored in the z-buffer).
- If not, we ignore the face.
- However, a pixel-by-pixel approach is taken.
Do it by scan line

- Initialize z-buffer to maximum distance from viewpoint
- Generate incremental equations for z for the polygons
- Each polygon can be represented by a plane equation of the form $ax + by + cz + d = 0$
- Given two points on the plane, we can write the plane equation in differential form: $a\Delta x + b\Delta y + c\Delta z = 0$
Do it by scan line (cont.)

- A scan line corresponds to a line of constant $y$ so $\Delta y = 0$
- As we move across the scan line $\Delta x$ will remain constant (we are taking unit steps, corresponding to equal increments in window coordinates)
- So, $\Delta z = - (a/c) \Delta x$
- Note that this value only has to be computed once for each polygon
Depth sort

- An object space approach to hidden surface removal
- A variant of the painter’s algorithm (where the farthest thing is painted first, then the next closest, etc.)
- Sort polygons by farthest $z$ value
- If the minimum $z$ for a polygon is greater than the maximum $z$ for all other polygons, it can be painted first
Depth sort (cont.)

- Otherwise, for polygons with $z$ extents that overlap the one with largest $z$:
  - test $x$ and $y$ extents; if the $x$ or $y$ extents do not overlap the polygons do not obscure each other and the farthest one can be painted (if it survives all other overlapping $z$ polygons)
  - if the farthest $z$ polygon lies entirely on the opposite side of the plane (from the viewpoint) of the polygon being tested it can be painted (if it survives all other overlapping $z$ polygons)
Depth sort (cont.)

- if the projections of the polygons onto the x-y plane do not overlap, the farthest z polygon can be painted (if it survives all other overlapping z polygons)
- otherwise, have intersecting polygons and need to determine the intersection details and break the polygons into pieces
Shadow buffers

- A variant of the z-buffer algorithm for generating shadows
- Initialize shadow buffer to maximum pseudodepth possible for each element
- Position a camera at the light source and rasterize each face in the scene, but only test the pseudodepth of each point
Rendering a face

- Render the face as usual, determining:
  - Pseudodepth from the light source
  - Index location in shadow buffer
  - Value stored in shadow buffer

- If shadow buffer value is less than pseudodepth
  the point is in shadow; determine its color using only ambient light

- Otherwise point is not in shadow and set its color using ambient, diffuse, and specular light
Features of texture mapping

- Makes images of 3-D objects more interesting and apparently more complex
- Diminishes the ‘shiny plastic’ effects of Phong shading
- Enables different objects to exhibit different surface properties (apart from color)
Reality vs. computer graphics

- Textures on real objects modulate both the surface and the color.
- Most texture mappings in computer graphics modulate just the color.
- Use repeating patterns or an image (called a texture map, and its individual elements are called texels).
Major considerations

- What attribute or parameter of the model or object is to be modulated to produce the desired textual effect?
- How is the texture mapping to be carried out?
  - A texture is defined in a texture domain
  - An object exists in world space
  - Need to define a mapping between the two
- Texture mapping requires special anti-aliasing treatment because it tends to produce worse aliasing artifacts than other techniques associated with image synthesis
Things you can modulate for texture

- Surface color (diffuse reflection coefficient)
- Specular and diffuse reflection coefficients (environment mapping)
- Surface normal perturbation (bump mapping)
- Transparency
Implementations

- Mapping from texture space to object space (usually 2-D to 3-D) and then performing the viewing projection to screen space
- Mapping directly from texture space to screen space (usually 2-D to 2-D)
  - Most attention has been paid to his method
  - An inverse mapping is commonly used, where the screen space is uniformly sampled and the inverse or pre-image of a pixel in texture space is formed
Procedural texture mapping

- Depending on the mapping and the object, many texels may map to one pixel or many pixels may map to one texel.
- Can circumvent the mapping problems by employing procedural texture mapping.
- For example, consider carving an object out of a solid block of material.
- Given the points on the object’s surface, we can determine its texture by applying a mapping function to the coordinates.
- An advantage to this approach is that it can be applied to objects of arbitrary complexity in a ‘coherent’ fashion (no discontinuities in texture will appear on the object).
Example program

- Program `texgen.c` displays a texture mapped teapot
- First, need to make the texture image
- For a 2-D image you need a rectangular array no less than 64x64, with each dimension being a power of 2
- If you are going to specify r,g,b values for each texel, then each element of the rectangular array must contain 3 ints between 0 and 255
- Study the `makeImage()` routine
Texture generation methods

- `glPixelStorei(GL_UNPACK_ALIGNMENT, 1);`
  - specifies the alignment requirement for the start of each pixel row in memory
  - 1 = byte, 2 = even-numbered bytes, 4 = word, 8 = double word
Texture generation methods (cont.)

- `glTexEnvi (GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_MODULATE);`
  - specifies how texture values are interpreted when a fragment is textured
  - first parameter must be `GL_TEXTURE_ENV` or `GL_TEXTURE_FILTER_CONTROL`
  - second parameter depends on first; here it is `GL_TEXTURE_ENV_MODE`
  - third parameter is one of `GL_MODULATE`, `GL_DECAL`, `GL_BLEND`, `GL_ADD`, `GL_COMBINE` or `GL_REPLACE`
Texture generation methods (cont.)

- `glTexParameteri (GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);`
  - sets texture parameters
  - first parameter indicates if texture is 1D, 2D, or 3D
  - second parameter identifies the texture parameter being affected
  - third parameter specifies the parameter’s value (in the case of wrapping it can be `GL_CLAMP`, `GL_CLAMP_TO_EDGE`, `GL_CLAMP_TO_BORDER`, `GL_REPEAT`, or `GL_MIRRORED_REPEAT`)
Texture generation methods (cont.)

- `glTexParameteri (GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);`
  - controls what happens when a pixel maps to a texture area less than or equal to 1 texel
  - `GL_NEAREST` returns the texel closest (in distance) to center of pixel being textured
  - `GL_LINEAR` returns the weighted average of the four texels closest to the center of the pixel being textured
Texture generation methods (cont.)

- `glTexImage2D (GL_TEXTURE_2D, 0, 3, ImageLength, ImageWidth, 0, GL_RGB, GL_UNSIGNED_BYTE, Image);`
  - specifies a 2-D texture image
  - level (0) indicates level of detail number (0 is base; used for mipmaps)
  - components (3) specifies # of colors
  - length and width must be powers of 2 plus 2*border
  - border (0) specifies border width (0 or 1)
  - format specifies the format of pixel data
  - type specifies the type of pixel data
  - Image is a pointer to the image data in memory
Texture generation methods (cont.)

- `glTexGeni (GL_S, GL_TEXTURE_GEN_MODE, GL_OBJECT_LINEAR);`
- `glTexGenfv (GL_S, GL_OBJECT_PLANE, sgenparams);`
  - provides automatic texture coordinate generation
  - first parameter indicates which coordinate
  - second parameter indicates coordinate generation function or its parameters
  - third parameter indicates which generation mode or the coefficients for the linear combination of the object coordinates
Image files for textures

- Can also use image files for texture maps
- Example program also shows a targa file *(stone.tga)* being used as a texture map